IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	Samuel D. Dull III, et al.	:	Date: June 8, 2006
Group Art Unit:	2192	:	IBM Corporation
Examiner:	A. Fowlkes	:	Intellectual Property Law
Serial No.:	09/821,920	:	Dept 917, Bldg 006-1
Filed:	March 20, 2001	:	3605 Highway 52 North
Title:	METHOD AND APPARATUS FOR INSTALLING AND UPGRADING AN APPLICATION IN A COMPUTER SYSTEM	:	Rochester, MN 55901

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 223313-1450

APPEAL BRIEF IN SUPPORT OF APPEAL FROM THE PRIMARY EXAMINER TO THE BOARD OF APPEALS

Sir:

This is an appeal of a Final Rejection under 35 U.S.C. §102(b) of claims 9, 15, 16 and 19 of Application Serial No. 09/821,920, filed March 30, 2001. This brief is submitted pursuant to a Notice of Appeal filed April 10, 2006, as required by 37 C.F.R. §1.192.

1. Real Party in Interest

International Business Machines Corporation of Armonk, NY, is the real party in interest. The inventors assigned their interest as recorded on June 26, 2001 on Reel 011950, Frame 02735

2. Related Appeals and Interferences

There are no related appeals nor interferences pending with this application.

3. Status of Claims

Claims 1-5 and 7-26 are pending and stand finally rejected. Claim 6 has been cancelled. Appellants contest the rejection of claims 9, 15, 16 and 19 only in this appeal. The claims on appeal are set forth in the Appendix of Claims

4. Status of Amendments

Following Final Rejection on November 30, 2005, appellants submitted a proposed Amendment on January 30, 2006, intended to reduce issues for appeal. The amendment cancelled certain claims, and incorporated the limitations of dependent claims 15 and 19 in independent claims 12 and 18, respectively, from which they depended. I.e., amended independent claims 12 and 18 were of exactly the same scope as previous dependent claims 15 and 19, respectively, but in independent form. By Advisory Action dated March 13, 2006, the Examiner refused to enter this amendment on the grounds that it would require a new search.

5. Summary of Claimed Subject Matter

The invention herein relates to software maintenance, and particularly to automatically installing upgrades for software installed in multiple systems. A computer system being upgraded includes a script processing program for executing upgrade scripts

(Spec. p. 4, lines 8-9; 17-18; p. 6, lines 16 - p. 7, line 3; Fig. 1, feature 190). An upgrade object in the form of a script having a set of instructions executable by the script processing program is created by a system administrator or similar person responsible for maintaining the software (Spec. p. 4, lines 13-16; p. 5, lines 6-10; p. 6, lines 11-18; p. 7, lines 6-10; Fig. 2, feature 200). The upgrade object is not directly executable code, but a set of instructions understandable to the script processing program, which reads the instructions and performs them (Spec. p. 4, lines 9-16; p. 6, lines 11-18; Fig. 2, feature 200). These instructions specify the steps required to perform the upgrade, much as might be performed manually by experienced personnel, or by custom-written and compiled executable code, using conventional techniques (Spec. p. 7, lines 11-20, p. 13, lines 5-18). The upgrade object is then distributed to multiple systems (e.g., electronically), and is invoked from the script processor to perform the upgrade (Spec. p. 8, lines 1-15; p. 9, lines 13-20; Fig. 3, step 301) .

6. Grounds of Rejection To Be Reviewed on Appeal

Claims 1-5 and 7-26 are finally rejected under 35 U.S.C. §102(b) as anticipated by Shrader et al. (US 5,870,611). Appellants appeal the rejection only with respect to claims 9,15, 16 and 19. The only substantial issues in this appeal are whether claims 9, 15, 16 and/or 19 are either anticipated by *Shrader*, or prima facie obvious over *Shrader*

7. Argument

Appellants contend that the Examiner failed to establish adequate grounds of rejection for the following reasons:

I. The Examiner improperly ignored significant claim limitations which were not disclosed in *Shrader*, specifically, the use of an upgrade object in the form of a script, which is executed on the computer system being upgraded by a separate script processor.

II. *Shrader* does not teach or suggest appellants' claimed invention, because *Shrader* discloses construction of a self-contained "installation object" for performing a program installation, and teaches away from appellants' claimed technique.

Overview of Invention

A brief overview of appellants' invention in light of existing art will be helpful in appreciating the issues herein. Maintaining software in numerous and diverse customer installations is a major problem for software vendors and their customers. Distribution of upgrade code can be performed by various means, e.g., electronic means via the Internet or other transmission mechanisms, distribution of physical media in the form of CD-ROMs or similar, etc. But distribution is only one step in upgrading a computer program. Installation of the upgrade code on the user's system is a separate, and often much more difficult, problem. Each user's system may be different, employing different hardware configurations, different operating systems, different software applications, and so forth. Each of multiple software applications or operating systems may further be at any of different levels and versions. In some cases, old versions have to be removed, system resources re-allocated, configuration tables updated, required run time libraries or other supporting modules installed, etc.

Historically, software has generally been installed manually by trained personnel. A significant level of expertise was required to install software on different systems, and the installer had to be trained to deal with multiple possible configurations and applications. As the number of computer systems has grown, it has become increasingly undesirable to install software upgrades manually. Manual installation is still used, particularly for large systems or installations which are relatively small in number, but it is obvious that this technique becomes very expensive and burdensome as the number of installations increases.

It is alternatively possible to provide customized automated installation programs, which are distributed to the users with the application upgrades, and automatically perform the installation process. This approach is commonly used for mass produced software, such as software for personal computers, in which it is assumed that many of the users have very little expertise. Generally, a user simply invokes the automated installation program, and the installation program does the rest. In some cases, the installation program may request installation parameters or other information from the user, but generally the process is automated to avoid requiring special knowledge or expertise of the user.

Conventional customized automated installation programs are just that, i.e., they are ***executable programs*** which perform an installation function automatically. Because such automated installation programs must be designed to accommodate a wide variety of system configurations and to execute with a minimum of end user input, they must automatically take into account these permutations and act accordingly. Such installation programs are often very complex. Development of such complex installation programs is a very involved process, like developing any complex application software. As in the case of the application program itself, the developer may use any of various known development tools for developing the installation program, and will compile it to executable code on a development system, before distributing it to the various end users. The installation program itself may occupy considerable memory volume, and transmission of such code by electronic means consumes considerable bandwidth of the transmission medium. When multiplied by the numerous upgrades and the number of systems to be upgraded, this consumption of transmission bandwidth can become a significant problem.

Applicants' invention provides an alternative upgrade process, which is suitable to certain environments. In accordance with applicants' preferred embodiment, the upgrade process is controlled by upgrade objects in the form of scripts. ***An upgrade object (script) is not an executable program***, but a set of commands or instructions which are executed by a ***separate script processor***. The script processor is distributed in advance, preferably with the originally installed computer program, but only need be distributed once. Thus, the script processor is not required to be distributed with each upgrade, effectively reducing the transmission bandwidth required to support upgrades. The script processor is executable code which reads the script contained in the update object and performs the upgrade according to the script. I.e., the script processor is essentially a compiler or interpreter which interprets the script commands and performs the functions specified therein. Each time an upgrade is required, a system administrator or similar person creates an upgrade object, i.e. a script, setting forth the operations to be performed by the script processor. These operations can be things installation tasks which the script processor does completely automatically , or can be prompts to a user to do something manually (e.g. "Insert diskette 'B'").

Applicants' invention therefore provides an alternative which is much easier to develop than customized code, and need not account for all possible configurations. At the same time, it achieves some degree of automation and relieves the user of the burden of knowing how to install the upgrade himself. A system administrator may create different upgrade objects for each of multiple common configurations which he supports. The upgrade object, being separate from the executable code (in the script processor) which actually performs the upgrade functions, is relatively compact and easy to transmit via electronic means or other.

I. The Examiner improperly ignored significant claim limitations which were not disclosed in *Shrader*, specifically, the use of an upgrade object in the form of a script, which is executed on the computer system being upgraded by a separate script processor.

In order to support a rejection for anticipation, each and every element of the claimed invention must be shown in a single prior art reference. Appellants' claims are not anticipated by *Shrader* because, inter alia, *Shrader* fails to teach the use of an upgrade object ***in the form of a script***, which contains not executable instructions but commands to a ***separate script processor***, the separate script processor executing the commands to perform the upgrade operations.

Shrader discloses a technique for building a customized automated installation program for installing software, which *Shrader* calls an "installation plan object". The installation plan object is built by a software developer on a development system, using object-oriented programming techniques. *Shrader* starts with an empty "installation plan object", which is a template for constructing the installation program. The developer then selects "child objects" from a library of such objects for inclusion in the installation plan object, as required to customize the installation. The developer also selects an application object, representing the application to be installed, and a group object representing the systems upon which the application will be installed, for inclusion in the installation plan object. The completed installation plan object either contains the executable installation code for installing the application or a reference to it.[1]

It should be understood that ***Shrader is describing a process that takes place at the software developer***. I.e., the software developer, in order to develop an installation program, builds the program from *Shrader*'s container object and child objects, using object-oriented programming concepts. However, the installation program, once built, is ***executable code***. This code is then distributed in the conventional fashion, and executed on the individual customer systems to install the software.

Shrader does not anticipate applicants' claims because, inter alia, *Shrader* does not teach the use of a script processor, separate from the upgrade object, which translates script in the upgrade object to execute the upgrade or installation.

Appellants' representative claim 9, a dependent claim, would contain the following recitations if written in independent form:

[1] Among other things, the installation object contains "code server objects", described as follows:

> FIG.6 depicts the code server container object 62 with a code server object 350 and an application image object 400. The code server object represents a physical file server machine which store the application images and which is to be accessed by client workstations during an installation, configuration, reinstall or removal of application software. The code server also contains directories to the various command and response files needed to perform the remote installation. The Code Server object 350 encapsulates the data and methods needed to discover where installable code images are located and to build network file system attach commands for those code images. It can also be used to store generated response files and output log files. It is a container for application images.

> [9]. A method of upgrading a computer program on a computer system, the computer program including an instruction processing module, the method comprising:
>
> receiving an upgrade object associated with the computer program, the upgrade object including an *instruction set adapted for use by the instruction processing module* to upgrade the computer program; and
>
> executing the instruction set with the instruction processing module; [from independent claim 1]
>
> wherein the *instruction set comprises a script* [from dependent claim 8].
>
> wherein the *instruction processing module is adapted to compile and execute the script* [from dependent claim 9] [emphasis added]

Dependent claims 15 and 19 vary in their language, but both recite the use of an upgrade object having upgrade instructions in the form of a script, which are compiled and executed by a script processor.

Appellant's independent claim 16 recites:

> 16. A method of upgrading a computer program on a computer system, comprising:
>
> (i) installing a computer program on the computer system, the *computer program including a script processing module*;
>
> (ii) receiving an upgrade object associated with the computer program, the upgrade object including a script adapted for use by the script processing module and a prerequisite field containing one or more prerequisites; and
>
> (iii) instructing the computer program to process the installation object, whereby the upgrade object causes the computer system to:
>
> determine if the one or more prerequisites have been met; and
>
> *instruct the script processing module to execute the script*; and
>
> report that the script has been executed [emphasis added]

Thus, although claim 16 does not explicitly recite compilation of the script by the script processing module, it recites that the script processing module is installed with the computer program (i.e, separate from and before receiving the upgrade object), and that the script processing module in the computer program determines if prerequisites are met and executes the script.

As a fundamental matter, it should be understood that ***Shrader describes a process which takes place at the software developer***, not at the system being upgraded. All of the choices are made by the developer to construct the installation object. Just what is this installation object? It is a container having various objects in it, but in particular it contains the installation programs in the form of executable code or contains references to one or more servers from which the executable code can be obtained. It is manifestly not a script, i.e., it is not a series of instructions meant to be executed by a script processor.

The thrust of *Shrader* is the creation of the installation object at the software developer, and it is this process of creating the installation object which they regard as their invention. Accordingly, *Shrader* does not describe in great detail what occurs at the system being upgraded. The installation object is intended to be a self-contained object, having everything that is necessary for installation of ths software (or being able to access it independently from a server).

Shrader contains no disclosure whatsoever of a script processor or equivalent which is included with the computer program being upgraded, installed in the system in which the upgrade is being installed, and which executes a script in the upgrade object.

Shrader does not explicitly discuss compilation of the installation programs. However, to the extent that such programs are compiled (as they presumably are), they are compiled in the development system, before the code ever reaches the system in which the application is to be installed.

It is apparently the Examiner's position that *Shrader*'s operating system is the equivalent of an instruction processing program (script processor).[2] Appellants challenge this reading. An operating system is not a compiler, interpreter or similar program which converts instructions in a script into executable form. An operating system does not "compile and execute" a script, as recited in claims 9, 15 and 19, nor does it "execute" a script, as recited in claim 16. In general, an operating system provides certain low-level functions which allocate system resources to processes, the processes executing the executable code.[3]

At a sufficiently high level of abstraction, an operating system resides at a lower level than application programming code and therefore helps it execute. But it is improper to abstract an invention to a high level of abstraction, and then find the invention in some prior art reference. It is the claim language which must govern a claim reading, and determines whether the reference anticipates the invention. Appellants claim that an entity (variously called a "script processor", "script processing module" or "instruction processing module"), which is installed with the program being upgraded or otherwise separately from the upgrade module, and which resides with that program in the same system, ***executes a script to perform the upgrade***. Only at the highest possible level of abstraction can an operating system be said to "execute" anything, and even at such a high level of abstraction,

[2] See Office Action of November 30, 2005, paragraph 7.

[3] Not all operating systems are identical or provide identical function. There are indeed some operating systems having built-in interpreters or compilers, which can generate executable code from higher level instructions or commands. But *Shrader* does not disclose that capability it their operating system, and in particular does not disclose or suggest that the operating system has the capability to execute scripts in the installation object.

the operating system does not compile a script, and does not execute an upgrade object of scripts to upgrade software resident in the system.

Finally, the Examiner notes that *Shrader* discloses an install plan object may contain an "Install Script" attribute identifying a code server object and directory path of install script files. The mere fact that *Shrader* uses the word "script" does not amount to a disclosure of appellant's invention. For the same reasons stated above, there is no disclosure in *Shrader* of a script processor which executes instructions in the form of a script to perform the upgrade, the script processor being previously installed with the program being upgraded or otherwise separately from the upgrade.

Since *Shrader* fails to teach at least one essential claim limitation of each of the claims at issue, the claims are not anticipated by *Shrader*, and the rejection of the claims on this ground was erroneous.

II. ***Shrader*** **does not teach or suggest appellants' claimed invention, because** ***Shrader*** **discloses construction of a self-contained "installation object" for performing a program installation, and teaches away from appellants' claimed technique.**[4]

Shrader discloses a technique for installing software, in which a software developer incorporates the application code to be installed and certain programming code which perform the installation into a self-contained installation object. This self-contained installation object does not require a separate script processor for performing the

[4] Generally, a rejection for anticipation under 35 U.S.C. §102 may be deemed to include an implied or "subsumed" single reference rejection for obviousness under 35 U.S.C. §103. The subsumed obviousness rejection is addressed here.

installation, since it contains (or references) the executable code which performs these steps. Accordingly, there is no motivation shown in *Shrader* to provide a script processing module as part of a program being upgraded, and a separate update script which executed by the script processing module.

Far from suggesting appellants' claimed invention, *Shrader* teaches away from it. *Shrader* discloses a technique which is based on executable installation code modules for performing installation tasks. The developer has some freedom to pick and choose those modules to be included in or referenced by the installation object, but the modules remain executable installation programs. Thus, *Shrader*'s technique may be viewed as an alternative to appellant's technique of using a script processor in the program being upgraded, and an upgrade object of script instructions executable by the script processor. There is nothing is *Shrader* that would teach or suggest the use of such a script processor.

8. Summary

Appellant discloses and claims a novel technique for upgrading software, in which a program to be upgraded contains a script processor (or equivalent), and a system administrator or other person creates an upgrade object containing instructions in the form of a script, which are executed by the script processor. *Shrader* discloses an essentially different technique for assembling components of an installation object in an object-oriented environment, in which the assembled installation object contains or references executable installation programs. *Shrader* neither teaches nor suggests the key limitations of a ***script processor*** (or equivalent), installed separately from and prior to the upgrade object in the system containing the program to be upgraded, which (compiles and) ***executes script*** in the upgrade object.

For all the reasons stated herein, the rejections for obviousness were improper, and appellant respectfully requests that the Examiner's rejections of the claims be reversed.

Date: June 8, 2006

Respectfully submitted,

SAMUEL D. DULL III, et al.

By [illegible]
Roy W. Truelson, Attorney
Registration No. 34,265

(507) 202-8725

APPENDIX OF CLAIMS

For completeness, all pending claims are contained in the appendix. However, as explained above, appellant appeals the rejection only of claims 9, 15, 16 and 19. The claims for which the rejection is not appealed are indicated.

1. (Rejection Not Appealed) A method of upgrading a computer program on a computer system, the computer program including an instruction processing module, the method comprising:

receiving an upgrade object associated with the computer program, the upgrade object including an instruction set adapted for use by the instruction processing module to upgrade the computer program; and

executing the instruction set with the instruction processing module.

2. (Rejection Not Appealed) The method of claim 1, further comprising instructing the instruction processing module to execute the instruction set.

3. (Rejection Not Appealed) The method of claim 1, wherein the upgrade object is associated with one or more prerequisites; and further comprising determining if the one or more prerequisites have been met.

4. (Rejection Not Appealed) The method of claim 1, further comprising reporting that the instruction set has been executed.

5. (Rejection Not Appealed) The method of claim 1, wherein the upgrade object is suitable for transmission by electronic mail.

6. (Cancelled)

7. (Rejection Not Appealed) The method of claim 1, wherein the instruction set comprises binary instructions.

8. (Rejection Not Appealed) The method of claim 1, wherein the instruction set comprises a script.

9. The method of claim 8, wherein the instruction processing module is adapted to compile and execute the script.

10. (Rejection Not Appealed) The method of claim 1, wherein the upgrade object requires the instruction processing module to be executed by the computer system.

11. (Rejection Not Appealed) The method of claim 1, wherein the upgrade object is not independently executable.

12. (Rejection Not Appealed) A method of upgrading a computer program on a computer system, the computer program including a script processor, the method comprising:

creating an upgrade object associated with the computer program, the upgrade object including an instruction set adapted for use by the script processor to upgrade the computer program; and

transmitting the upgrade object to the computer system; and

instructing an end user to execute the instruction set with the script processor.

13. (Rejection Not Appealed) The method of claim 12, wherein the upgrade object presents the end user with instructions to perform a task.

14. (Rejection Not Appealed) The method of claim 13, wherein the upgrade object prompts the end user to indicate that the task has been performed.

15. The method of claim 12, wherein the instruction set comprises a script and wherein the script processor is adapted to compile and execute the script.

16. A method of upgrading a computer program on a computer system, comprising:

(i) installing a computer program on the computer system, the computer program including a script processing module;

(ii) receiving an upgrade object associated with the computer program, the upgrade object including a script adapted for use by the script processing module and a prerequisite field containing one or more prerequisites; and

(iii) instructing the computer program to process the installation object, whereby the upgrade object causes the computer system to:

determine if the one or more prerequisites have been met; and

instruct the script processing module to execute the script; and

report that the script has been executed.

17. (Rejection Not Appealed) A computer program product, comprising:

(a) an upgrade object configured to upgrade a software program having an instruction processing module, the upgrade object including an instruction set capable of causing the instruction processing module to perform one or more upgrade tasks; and

(b) a tangible signal bearing media bearing the upgrade object.

18. (Rejection Not Appealed) A method of installing a computer program into an instruction processing environment on a computer system, the instruction processing environment including an instruction processing module, the method comprising:

receiving an installation object associated with the computer program, the installation object including an instruction set adapted for use by the instruction processing module to install the computer program into the instruction processing environment; and

executing the instruction set with the instruction processing module.

19. The method of claim 18, wherein the instruction set comprises a script and wherein the instruction processing module is adapted to compile and execute the script.

20. (Rejection Not Appealed) The method of claim 18, wherein the installation object is associated with one or more prerequisites; and further comprising determining if the one or more prerequisites have been met.

21. (Rejection Not Appealed) The method of claim 18, further comprising reporting that the instruction set has been executed.

22. (Rejection Not Appealed) The method of claim 18, wherein the installation object comprises a script suitable for transmission by electronic mail.

23. (Rejection Not Appealed) A method of installing a computer program into an instruction processing environment on a computer system, the instruction processing environment including an instruction processing module, the method comprising:

creating an installation object associated with the computer program, the upgrade object including an instruction set adapted for use by the instruction processing module to install the computer program into the instruction processing environment;

transmitting the installation object to the computer program; and

instructing an end user to execute the instruction set with the instruction processing module.

24. (Rejection Not Appealed) The method of claim 23, wherein the installation object presents the end user with instructions to perform a task.

25. (Rejection Not Appealed) The method of claim 24, wherein the installation object prompts the end user to indicate that the task has been performed.

26. (Rejection Not Appealed) A method for upgrading a computer program on a computer system, the computer program including an instruction processing program module, the method comprising configuring the computer system to perform the method of claim 1.

APPENDIX OF EVIDENCE

No evidence is submitted.

APPENDIX OF RELATED PROCEEDINGS

There are no related proceedings.